University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: March 3, 2025

Group Number: 56

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Nicholas Huang	54394416	n0t1v	nhuangra@gmail.com
Louis Luo	97525331	j4m1n	louisjyluo@gmail.com
Perry Huang	70184767	v7l9t	perryhp2012@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

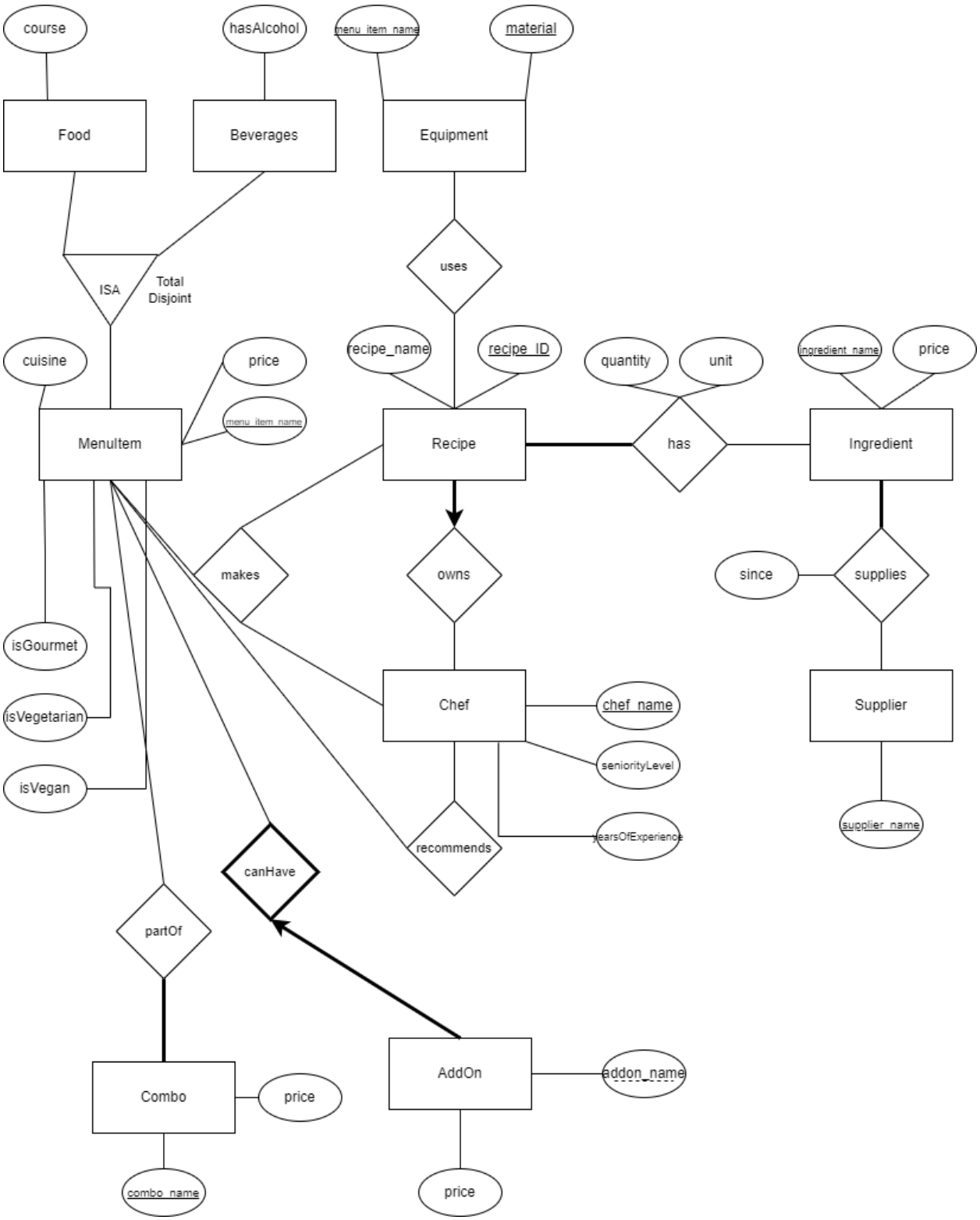
In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

No AI tools have been used during the creation of this milestone.

Project Description

This application is for a restaurant test kitchen. It helps with menu development for restaurants. The domain is contained within the entirety of the kitchen plus the suppliers. The application also contains the recipes for each menu item which are provided by the chefs. We have chefs who develop recipes which take Ingredient, and those Ingredient will all be supplied by suppliers. Each recipe requires equipment to be made into menu items. The recipes are used by the chefs and they make menu items, and those menu items are either foods or drinks. Each menu item can also be a part of a combo, or have add-ons such as fries or drinks.

ER Diagram


course
hasAlcohol
menu_item_name
material
Food
Beverages
Equipment
ISA
Total
Disjoint
uses
cuisine
price
menu_item_name
recipe_name
recipe_ID
quantity
unit
ingredient_name
price
MenuItem
Recipe
has
Ingredient
makes
owns
since
supplies
isGourmet
isVegetarian
isVegan
Chef
chef_name
seniorityLevel
yearsOfExperience
recommends
Supplier
supplier_name
canHave
partOf
Combo
price
combo_name
AddOn
addon_name
price

FIX TO ER DIAGRAM:

Added Total and Disjoint to ISA relation
name -> supplier_name
name -> ingredient_name
name -> chef_name
name -> menu_item_name
name -> equipment_name
name -> combo_name
name -> addon_name
name -> recipe_name
ID -> recipe_ID
Added price to Addon
Added “since” to Supplies
Added “isGourmet,” “isVegetarian,” and “isVegan” to MenuItem
Added “seniorityLevel” and “yearsOfExperience” to Chef

We added the Total and Disjoint subclass relation on the ISA from MenuItem to Food and Beverage from the feedback from milestone 1. To ensure the FDs don’t get too confusing with all the different “name” primary keys, we refactored all the names and IDs. Then to add a few attributes that can be non-primary key FDs we added “isGourmet,” “isVegetarian,” and “isVegan” to MenuItem and added “seniorityLevel” and “yearsOfExperience” to Chef to ensure our BCNF decomposition can be a bit more challenging to decompose.

4. ER diagram to Relational Schema

Supplier(supplier_name:Supplier)
Ingredient(ingredient_name:Ingredient, price:Ingredient) (ingredient_name is not null)
Supplies(**supplier_name**:Supplier, **ingredient_name**:Ingredient, since:Supplies) (**ingredient_name**:Ingredient is not null and since:Supplies is not null)
RecipeOwns(recipe_ID:Recipe, **chef_name**: Chef, recipe_name: Recipe) (**chef_name**: Chef, recipe_name: Recipe is not null)
Has(**recipe_ID**:Recipe, **ingredient_name**:Ingredient, quantity:Has, unit:Has) (**ingredient_name**:Ingredient is not null)
Chef(chef_name:Chef, seniority: Chef, yearsOfExperience: Chef)
Recommends(**chef_name**:Chef, **menu_item_name**:MenuItem)
Equipment(equipment_name: Equipment, material: Equipment)
Uses(**equipment_name**: Equipment, **material**: Equipment, **recipe_ID**:Recipe)
Makes(**recipe_ID**:Recipe, **chef_name**:Chef, **menu_item_name**:MenuItem)

MenuItem(menu_item_name:MenuItem, cuisine: MenuItem, price: MenuItem, isGourmet: MenuItem, isVegetarian: MenuItem, isVegan: MenuItem) (price: MenuItem is not null, cuisine: MenuItem is not null)
Food(**menu_item_name**:MenuItem, course: Food)
Beverages(**menu_item_name**:MenuItem, hasAlcohol: Beverages)
AddOnCanHave(**menu_item_name**: MenuItem, addon_name: AddOn, price: AddOn)
Combo(combo_name:Combo, price: Combo) (price: Combo is not null)
PartOf(**combo_name**:Combo, **menu_item_name**: MenuItem)

5. Functional Dependencies (FDs)

ingredient_name: Ingredient –> price: Ingredient
supplier_name: Supplier, ingredient_name: Ingredient → since: supplies
recipe_ID: Recipe → recipe_name: Recipe, chef_name: Chef
recipe_ID: Recipe, ingredient_name: Ingredient → quantity: Has, unit: Has
recipe_ID: Recipe → menu_item_name: Menuitem
chef_name: Chef → seniority: Chef, yearsOfExperience: Chef
yearsOfExperience: Chef → seniority: Chef
recipe_ID: Recipe → equipment_name: Equipment, material: Equipment
menu_item_name: MenuItem → cuisine: MenuItem, price: MenuItem, isGourmet: MenuItem, isVegetarian: MenuItem, isVegan: MenuItem, course: Food, hasAlcohol: Beverages
price: MenuItem → isGourmet: MenuItem
isVegan: MenuItem → isVegetarian: MenuItem
addon_name: Addon -> price: AddOn
combo_name: Combo → price: Combo

6. Normalization (BCNF Decomposition)

FDs

ingredient_name: Ingredient –> price: Ingredient
supplier_name: Supplier, ingredient_name: Ingredient → since: supplies
recipe_ID: Recipe → recipe_name: Recipe, chef_name: Chef
recipe_ID: Recipe, ingredient_name: Ingredient → quantity: Has, unit: Has
recipe_ID: Recipe → menu_item_name: Menuitem
chef_name: Chef → seniority: Chef, yearsOfExperience: Chef
yearsOfExperience: Chef → seniority: Chef

recipe_ID: Recipe → equipment_name: Equipment, material: Equipment
menu_item_name: MenuItem → cuisine: MenuItem, price: MenuItem, isGourmet: MenuItem, isVegetarian: MenuItem, isVegan: MenuItem, course: Food, hasAlcohol: Beverages
price: MenuItem → isGourmet: MenuItem
isVegan: MenuItem → isVegetarian: MenuItem
addon_name: Addon -> price: AddOn
combo_name: Combo → price: Combo

Tables by relevant FDs:

Ingredient(ingredient_name:Ingredient, price:Ingredient) (ingredient_name is not null)
ingredient_name: Ingredient –> price: Ingredient

Supplies(**supplier_name**:Supplier, **ingredient_name**:Ingredient, since:Supplies)
(**ingredient_name**:Ingredient is not null and since:Supplies is not null)
supplier_name: Supplier, ingredient_name: Ingredient → since: supplies

Supplier(supplier_name:Supplier)
None

RecipeOwns(recipe_ID:Recipe, **chef_name**: Chef, recipe_name: Recipe,)
(**chef_name**: Chef is not null, recipe_name: Recipe is not null)
recipe_ID: Recipe → recipe_name: Recipe, chef_name: Chef

Has(**recipe_ID**:Recipe, **ingredient_name**:Ingredient, quantity:Has, unit:Has)
(**ingredient_name**:Ingredient)
recipe_ID: Recipe, ingredient_name: Ingredient → quantity: Has, unit: Has

Chef(chef_name:Chef, seniority: Chef, yearsOfExperience: Chef)
chef_name: Chef → seniority: Chef, yearsOfExperience: Chef
yearsOfExperience: Chef → seniority: Chef

Recommends(**chef_name**:Chef, **menu_item_name**:MenuItem)
None

Equipment(equipment_name: Equipment, material: Equipment)

None

Uses(**equipment_name**: Equipment, **material**: Equipment, **recipe_ID**:Recipe)
recipe_ID: Recipe → equipment_name: Equipment, material: Equipment

Makes(**recipe_ID**:Recipe, **chef_name**:Chef, **menu_item_name**:MenuItem)
recipe_ID: Recipe → menu_item_name: Menuitem

MenuItem(menu_item_name:MenuItem, cuisine: MenuItem, price: MenuItem, isGourmet: MenuItem, isVegetarian: MenuItem, isVegan: MenuItem) (price: MenuItem is not null, cuisine: MenuItem is not null)
menu_item_name: MenuItem → cuisine: MenuItem, price: MenuItem, isGourmet: MenuItem, isVegetarian: MenuItem, isVegan: MenuItem, course: Food, hasAlcohol: Beverages
price: MenuItem → isGourmet: MenuItem
isVegan: MenuItem → isVegetarian: MenuItem

Food(**menu_item_name**:MenuItem, course: Food)
menu_item_name: MenuItem → cuisine: MenuItem, price: MenuItem, isGourmet: MenuItem, isVegetarian: MenuItem, isVegan: MenuItem, course: Food, hasAlcohol: Beverages
price: MenuItem → isGourmet: MenuItem

Beverages(**menu_item_name**:MenuItem, hasAlcohol: Beverages)
menu_item_name: MenuItem → cuisine: MenuItem, price: MenuItem, isGourmet: MenuItem, isVegetarian: MenuItem, isVegan: MenuItem, course: Food, hasAlcohol: Beverages
price: MenuItem → isGourmet: MenuItem

AddOnCanHave(**menu_item_name**: MenuItem, addon_name: AddOn, price: AddOn)
addon_name: Addon -> price: AddOn

Combo(combo_name:Combo, price: Combo) (price: Combo is not null)
combo_name: Combo → price: Combo

PartOf(**combo_name**:Combo, **menu_item_name**: MenuItem)
None

Normalized BCNF Tables:

Chef(chef_name:Chef, seniority: Chef, yearsOfExperience: Chef)
chef_name: Chef → seniority: Chef, yearsOfExperience: Chef
yearsOfExperience: Chef → seniority: Chef (yearsOfExperience is not a superkey)

	X	->	b
chef_name: Chef	yearsOfExperience: Chef		seniority: Chef

ChefStatus(yearsOfExperience: ChefStatus ,seniority: ChefStatus)
ChefExperience(chef_name: ChefExperience, **yearsOfExperience**: ChefStatus)

MenuItem(menu_item_name:MenuItem, cuisine: MenuItem, price: MenuItem, isGourmet: MenuItem, isVegetarian: MenuItem, isVegan: MenuItem) (price: MenuItem is not null, cuisine: MenuItem is not null)
price: MenuItem → isGourmet: MenuItem (price is not a superkey)
isVegan: MenuItem → isVegetarian: MenuItem (isVegan is not a superkey)

	X	->	b
menu_item_name: MenuItem, cuisine: MenuItem, isVegetarian: MenuItem, isVegan: MenuItem	price: MenuItem		isGourmet: MenuItem

MenuItemPrice(price: MenuItemPrice, isGourmet: MenuItemPrice)
MenuItem'(price: MenuItem', menu_item_name: MenuItem', cuisine: MenuItem', isVegetarian: MenuItem', isVegan: MenuItem') (isVegan is not a superkey)

	X	->	b
name: MenuItem, cuisine: MenuItem, price: MenuItem,	isVegan: MenuItem		isVegetarian: MenuItem

MenuItemDiet(isVegan: MenuItemDiet, isVegetarian: MenuItemDiet)
MenuItemDetails(**isVegan**: MenuItemDiet, menu_item_name: MenuItemDetails, cuisine: MenuItemDetails, **price**: MenuItemPrice)

AddOnCanHave(**menu_item_name**: MenuItem, addon_name: AddOn, price: AddOn)
addon_name: Addon -> price: AddOn

	X	->	b
menu_item_name: MenuItem	addon_name: Addon		price: Addon

AddOnPrice(addon_name: AddOnPrice, price: AddOnPrice)
AddOnCanHave(**addon_name**: AddOnPrice, **menu_item_name**: MenuItemDetails)

Already in BCNF Tables:

Ingredient(ingredient_name:Ingredient, price:Ingredient) (ingredient_name is not null)
Supplies(**supplier_name**:Supplier, **ingredient_name**:Ingredient, since:Supplies)
(**ingredient_name**:Ingredient is not null and since:Supplies is not null)
Supplier(supplier_name:Supplier)
RecipeOwns(recipe_ID:Recipe, **chef_name**: ChefExperience, recipe_name: Recipe,)
(**chef_name**: ChefExperience, recipe_name: Recipe is not null)
Has(**recipe_ID**:Recipe, **ingredient_name**:Ingredient, quantity:Has, unit:Has)
(**ingredient_name**:Ingredient is not null)
Recommends(**chef_name**:ChefExperience, **menu_item_name**:MenuItemDetails)
Equipment(equipment_name: Equipment, material: Equipment)
Uses(**equipment_name**: Equipment, **material**: Equipment, **recipe_ID**:Recipe)
Makes(**recipe_ID**:Recipe, **chef_name**:ChefExperience,
menu_item_name:MenuItemDetails)
Food(**menu_item_name**:MenuItemDetails, course: Food)
Beverages(**menu_item_name**:MenuItemDetails, hasAlcohol: Beverages)
Combo(combo_name:Combo, price: Combo) (price: Combo is not null)
PartOf(**combo_name**:Combo, **menu_item_name**:MenuItemDetails)

Final Result

Ingredient(ingredient_name:Ingredient, price:Ingredient) (ingredient_name is not null)
Supplies(**supplier_name**:Supplier, **ingredient_name**:Ingredient, since:Supplies)
(**ingredient_name**:Ingredient is not null and since:Supplies is not null)
Supplier(supplier_name:Supplier)
RecipeOwns(recipe_ID:Recipe, **chef_name**: ChefExperience, recipe_name: Recipe,)
(**chef_name**: ChefExperience, recipe_name: Recipe is not null)
ChefStatus(yearsOfExperience: ChefStatus, seniority: ChefStatus)
ChefExperience(chef_name: ChefExperience, **yearsOfExperience**: ChefStatus)

Has(**recipe_ID**:Recipe, **ingredient_name**:Ingredient, quantity:Has, unit:Has)
(**ingredient_name**:Ingredient is not null)
MenuItemPrice(price: MenuItemPrice, isGourmet: MenuItemPrice)
MenuItemDiet(isVegan: MenuItemDiet, isVegetarian: MenuItemDiet)
MenuItemDetails(**isVegan**: MenuItemDiet, menu_item_name: MenuItemDetails,
cuisine: MenuItemDetails, **price**: MenuItemPrice)
Recommends(**chef_name**:ChefExperience, **menu_item_name**:MenuItemDetails)
Equipment(equipment_name: Equipment, material: Equipment)
Uses(**equipment_name**: Equipment, **material**: Equipment, **recipe_ID**:Recipe)
Makes(**recipe_ID**:Recipe, **chef_name**:ChefExperience,
menu_item_name:MenuItemDetails)
Food(**menu_item_name**:MenuItemDetails, course: Food)
Beverages(**menu_item_name**:MenuItemDetails, hasAlcohol: Beverages)
Combo(combo_name:Combo, price: Combo) (price: Combo is not null)
PartOf(**combo_name**:Combo, **menu_item_name**:MenuItemDetails)
AddOnPrice(addon_name: AddOnPrice, price: AddOnPrice)
AddOnCanHave(**addon_name**: AddOnPrice, **menu_item_name**: MenuItemDetails)

7. The SQL DDL statements

```
CREATE TABLE Ingredient (
        ingredient_name VARCHAR,
        price FLOAT,
        PRIMARY KEY(ingredient_name)
);

CREATE TABLE Supplier (
        supplier_name VARCHAR,
        PRIMARY KEY(supplier_name)
);

CREATE TABLE Supplies (
        supplier_name VARCHAR,
        ingredient_name VARCHAR,
        since DATE,
```

```
        PRIMARY KEY(supplier_name, ingredient_name),
        FOREIGN KEY(supplier_name) REFERENCES Supplier
                ON DELETE NO ACTION
                ON UPDATE CASCADE,
        FOREIGN KEY(ingredient_name) REFERENCES Ingredient
                ON DELETE NO ACTION
                ON UPDATE CASCADE
);
```

Explanation: Neither a supplier nor an ingredient should be able to be deleted if either of them are participating in the Supplies relationship because deleting them would entirely lose the information of where those ingredients were coming from. This is not okay because the recipes in the database depend on those ingredients. However, we use ON UPDATE CASCADE because it's okay to update the supplier and ingredient names as long as we keep the changes consistent across the DB.

```
CREATE TABLE ChefStatus (
        seniority VARCHAR,
        yearsOfExperience INTEGER,
        PRIMARY KEY(yearsOfExperience)
);

CREATE TABLE ChefExperience (
        chef_name VARCHAR,
        yearsOfExperience INTEGER,
        PRIMARY KEY(chef_name),
        FOREIGN KEY(yearsOfExperience) REFERENCES ChefStatus
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
);
```

Explanation: The number of years of experience a chef has shouldn't be able to be changed just because a database entry was deleted or updated. The chef will have that many years of experience regardless.

```
CREATE TABLE RecipeOwns (
        recipe_ID INTEGER,
        chef_name VARCHAR NOT NULL,
        recipe_name VARCHAR NOT NULL,
        PRIMARY KEY(recipe_ID),
        FOREIGN KEY(chef_name) REFERENCES ChefExperience
```

```
            ON DELETE SET NULL
            ON UPDATE CASCADE
);
```

Explanation: If a chef is deleted, perhaps because they no longer work there or because they requested to have their information erased, we don't want to have to also delete the recipes that they created unless we have to. So we just set the chef_name to null so that we can keep the recipes in the DB. If the chef's name changes, then we can just reflect that change by cascading it to keep it consistent.

```
CREATE TABLE Has (
    recipe_ID INTEGER,
    ingredient_name VARCHAR,
    quantity FLOAT,
    unit VARCHAR,
    PRIMARY KEY(recipe_ID, ingredient_name),
    FOREIGN KEY(recipe_ID) REFERENCES RecipeOwns
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY(ingredient_name) REFERENCES Ingredient
        ON DELETE NO ACTION
        ON UPDATE CASCADE
);
```

Explanation: If a recipe is deleted, we don't need to keep storing which ingredients it used. But if an ingredient is deleted, we don't want to allow that if a recipe is using that ingredient. For both, we want to keep updates consistent so we use CASCADE.

```
CREATE TABLE MenuItemPrice (
    price FLOAT NOT NULL,
    isGourmet BIT NOT NULL,
    PRIMARY KEY(price)
);

CREATE TABLE MenuItemDiet (
    isVegan BIT NOT NULL,
    isVegetarian BIT NOT NULL,
    PRIMARY KEY(isVegan)
);
```

```
CREATE TABLE MenuItemDetails (
        menu_item_name VARCHAR,
        cuisine VARCHAR NOT NULL,
        isVegan BIT NOT NULL,
        price FLOAT NOT NULL,
        PRIMARY KEY(menu_item_name),
        FOREIGN KEY(isVegan) REFERENCES MenuItemDiet
                ON DELETE NO ACTION
                ON UPDATE CASCADE,
        FOREIGN KEY(price) REFERENCES MenuItemPrice
                ON DELETE NO ACTION
                ON UPDATE CASCADE
);
```

Explanation: We can't lose the information about whether a menu item is vegan or not. We also can't lose the information about a menu item's price. But we can allow updates to cascade to keep everything consistent.

```
CREATE TABLE Recommends (
        chef_name VARCHAR,
        menu_item_name VARCHAR,
        PRIMARY KEY(chef_name, menu_item_name),
        FOREIGN KEY(chef_name) REFERENCES ChefExperience
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY(menu_item_name) REFERENCES MenuItemDetails
                ON DELETE CASCADE
                ON UPDATE CASCADE
);
```

Explanation: It's not super important to maintain recommendations in the DB. So if a chef is deleted, we can throw out their recommendations too. If a menu item is deleted, its recommendation also doesn't need to stay. And if either are updated, we just keep them consistent with CASCADE.

```
CREATE TABLE Equipment (
        equpiment_name VARCHAR,
        equipment_material VARCHAR,
        PRIMARY KEY(equpiment_name, equipment_material)
);
```

```
CREATE TABLE Uses (
        equipment_name VARCHAR,
        equipment_material VARCHAR,
        recipe_ID INTEGER,
        PRIMARY KEY(equipment_name, equipment_material, recipe_ID),
        FOREIGN KEY(equipment_name, equipment_material) REFERENCES
Equipment
                ON DELETE NO ACTION
                ON UPDATE CASCADE,
        FOREIGN KEY(recipe_ID) REFERENCES RecipeOwns
                ON DELETE CASCADE
                ON UPDATE CASCADE
);
```

Explanation: We don't allow equipment to be deleted if they're being used by a recipe. But if a recipe is deleted, then we don't need to keep track of which equipment it used. If either are updated, we just want to keep them consistent with CASCADE.

```
CREATE TABLE Makes (
        recipe_ID INTEGER,
        chef_name VARCHAR,
        menu_item_name VARCHAR,
        PRIMARY KEY(recipe_ID, chef_name, menu_item_name),
        FOREIGN KEY(chef_name) REFERENCES ChefExperience
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY(menu_item_name) REFERENCES MenuItemDetails
                ON DELETE CASCADE
                ON UPDATE CASCADE
);
```

Explanation: When a chef or menu item is deleted, we don't care about the makes relationship between them anymore. If either is updated, we can just keep them consistent with CASCADE as well.

```
CREATE TABLE Food (
        menu_item_name VARCHAR,
        course VARCHAR,
        PRIMARY KEY(menu_item_name),
        FOREIGN KEY(menu_item_name) REFERENCES MenuItemDetails
```

```
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```
Explanation: Since food is a menu item, if a menu item is deleted or updated we can just do the same to the corresponding food entry.

```
CREATE TABLE Beverages (
    menu_item_name VARCHAR,
    hasAlcohol BIT NOT NULL,
    PRIMARY KEY(menu_item_name),
    FOREIGN KEY(menu_item_name) REFERENCES MenuItemDetails
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```
Explanation: Since beverages are menu items, if a menu item is deleted or updated we can just do the same to the corresponding beverages entry.

```
CREATE TABLE AddOnPrice (
    addon_name VARCHAR,
    price FLOAT,
    PRIMARY KEY(addon_name)
);

CREATE TABLE AddonCanHave (
    menu_item_name VARCHAR,
    addon_name VARCHAR,
    PRIMARY KEY(menu_item_name, addon_name),
    FOREIGN KEY(menu_item_name) REFERENCES MenuItemDetails
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY(addon_name) REFERENCES AddOnPrice
        ON DELETE NO ACTION
        ON UPDATE CASCADE
);
```
Explanation: If a menu item is deleted, we don't care what addons it could have. But we can't allow addons to be deleted if they are already an option for a menu item. If either are updated, we just want to keep them consistent with CASCADE.

```
CREATE TABLE Combo (
        combo_name VARCHAR,
        price FLOAT NOT NULL,
        PRIMARY KEY(combo_name)
);

CREATE TABLE PartOf (
        combo_name VARCHAR,
        menu_item_name VARCHAR,
        PRIMARY KEY(combo_name, menu_item_name),
        FOREIGN KEY(combo_name) REFERENCES Combo
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY(menu_item_name) REFERENCES MenuItemDetails
                ON DELETE NO ACTION
                ON UPDATE CASCADE
);
```

Explanation: If a combo is deleted, we don't need to keep track of which menu items were part of it. But we can't allow menu items to be deleted if they're part of a combo. We do want to keep them consistent with CASCADE if either are updated.

8. INSERT

Ingredient

```
INSERT
INTO Ingredient
VALUES ('Garlic', 1.00)

INSERT
INTO Ingredient
VALUES ('Turkey',  35.00)

INSERT
INTO Ingredient
VALUES ('Caviar', 500.00)
```

INSERT
INTO Ingredient
VALUES (‘Onion’, 0.50)

INSERT
INTO Ingredient
VALUES (‘Potato’, 0.67)

Supplies

INSERT
INTO Supplies
VALUES (“Joel’s Orchard”, 'Apple', '01-03-2021')

INSERT
INTO Supplies
VALUES (‘Nicks Butcher Shop’, ‘Turkey’, ‘15-02-2019’)

INSERT
INTO Supplies
VALUES (“Jenny’s Farm”, ‘Potato’, ‘20-05-2024’)

INSERT
INTO Supplies
VALUES (“Jenny’s Farm”, ‘Onion’, ‘20-05-2024’)

INSERT
INTO Supplies
VALUES (“Joel’s Orchard”, ‘Blueberries’, ‘01-03-2021’)

INSERT
INTO Supplies
VALUES(“Queen Anica’s Palace”, ‘Caviar’, 19-02-2004)

Supplier

INSERT
INTO Supplier

VALUES (“Joel’s Orchard”)

INSERT
INTO Supplier
VALUES (“Jenny’s Farm”)

INSERT
INTO Supplier
VALUES (“Nick’s Butcher Shop”)

INSERT
INTO Supplier
VALUES (‘Happy Farm’)

INSERT
INTO Supplier
VALUES(“Queen Anica’s Palace”)

RecipeOwns

INSERT
INTO RecipeOwns
VALUES (1, ‘Perry’, ‘Garlic Beef’)

INSERT
INTO RecipeOwns
VALUES (2, ‘Louis’, ‘BigWay Hot Pot’)

INSERT
INTO RecipeOwns
VALUES (3, ‘Ryan’, ‘Roast Turkey’)

INSERT
INTO RecipeOwns
VALUES (4, ‘Gordon’, ‘Beef Wellington’)

INSERT
INTO RecipeOwns
VALUES (5, ‘William’, ‘Mushroom Pizza’)

Has

INSERT
INTO Has
VALUES (6, ‘Apple’, 3, ‘Ib’)

INSERT
INTO Has
VALUES (3, ‘Turkey’, 1, ‘item’)

INSERT
INTO Has
VALUES (5, ‘Mushroom’, 1, ‘Ib’)

INSERT
INTO Has
VALUES (5, ‘Dough’, 3, ‘Ib’)

INSERT
INTO Has
VALUES (1, ‘Garlic’, 8, ‘item’)

ChefStatus

INSERT
INTO ChefStatus
VALUES (‘master’, 10)

INSERT
INTO ChefStatus
VALUES (‘apprentice’, 1)

INSERT
INTO ChefStatus
VALUES (‘master’, 20)

INSERT
INTO ChefStatus
VALUES (‘novice’, 2)

```
INSERT
INTO ChefStatus
VALUES ('beginner', 0)
```

ChefExperience

```
INSERT
INTO ChefExperience
VALUES ('Ryan', 10)

INSERT
INTO ChefExperience
VALUES ('Perry', 1)

INSERT
INTO ChefExperience
VALUES ('Gordon', 20)

INSERT
INTO ChefExperience
VALUES ('William', 2)

INSERT
INTO ChefExperience
VALUES ('Louis', 0)
```

Recommends

```
INSERT
INTO Recommends
VALUES ('Ryan', 'BigWay Hot Pot')

INSERT
INTO Recommends
VALUES ('Gordon', 'Beef Wellington')
```

INSERT
INTO Recommends
VALUES (‘Perry’, ‘Garlic Beef’)

INSERT
INTO Recommends
VALUES (‘Ryan’, ‘Pepperoni Pizza’)

INSERT
INTO Recommends
VALUES (‘Perry’, ‘Roast Turkey’)

Equipment

INSERT
INTO Equipment
VALUES (‘Spatula’, ‘Wood’)

INSERT
INTO Equipment
VALUES (‘Oven’, ‘Electronic’)

INSERT
INTO Equipment
VALUES (‘Pot’, ‘Metal’)

INSERT
INTO Equipment
VALUES (‘Whisk’, ‘Metal’)

INSERT
INTO Equipment
VALUES (‘Pan’, ‘Metal’)

Uses

INSERT
INTO Uses

VALUES (‘Pan’, ‘Metal’, ‘Garlic Beef’)

INSERT
INTO Uses
VALUES (‘Oven’, ‘Electronic’, ‘Mushroom Pizza’)

INSERT
INTO Uses
VALUES (‘Pot’, ‘Metal’, ‘BigWay Hot Pot’)

INSERT
INTO Uses
VALUES (‘Oven’, ‘Electronic’, ‘Roast Turkey’)

INSERT
INTO Uses
VALUES (‘Pan’, ‘Metal’, ‘Beef Wellington’)ERT

Makes

INSERT
INTO Makes
VALUES (1, ‘Louis’, ‘Garlic Beef’)

INSERT
INTO Makes
VALUES (2, ‘Louis’, ‘BigWay Hot Pot’)

INSERT
INTO Makes
VALUES (3, ‘Gordon’, ‘Roast Turkey’)

INSERT
INTO Makes
VALUES (4, ‘Gordon’, ‘Beef Wellington’)

INSERT
INTO Makes

VALUES (5, ‘Perry’, ‘Mushroom Pizza’)

MenuItemPrice

INSERT
INTO MenuItemPrice
VALUES (10.00, 0)

INSERT
INTO MenuItemPrice
VALUES (25.00, 0)

INSERT
INTO MenuItemPrice
VALUES (45.00, 0)

INSERT
INTO MenuItemPrice
VALUES (155.67, 1)

INSERT
INTO MenuItemPrice
VALUES (20.00, 1)

INSERT
INTO MenuItemPrice
VALUES (6.50, 0)

INSERT
INTO MenuItemPrice
VALUES (7.00, 0)

INSERT
INTO MenuItemPrice
VALUES (18.00, 1)

INSERT
INTO MenuItemPrice

```
VALUES (3.25, 0)

INSERT
INTO MenuItemPrice
VALUES (1.50, 0)
```

MenuItemDiet

```
INSERT
INTO MenuItemDiet
VALUES (0, 0)

INSERT
INTO MenuItemDiet
VALUES (0, 0)

INSERT
INTO MenuItemDiet
VALUES (0, 0)

INSERT
INTO MenuItemDiet
VALUES (0, 0)

INSERT
INTO MenuItemDiet
VALUES (0, 0)

INSERT
INTO MenuItemDiet
VALUES (1, 1)

INSERT
INTO MenuItemDiet
VALUES (0, 1)

INSERT
INTO MenuItemDiet
VALUES (0, 0)
```

```
INSERT
INTO MenuItemDiet
VALUES (1, 1)

INSERT
INTO MenuItemDiet
VALUES (1, 1)
```

MenuItemDetails

```
INSERT
INTO MenuItemDetails
VALUES ('Garlic Beef', 'Chinese', 0, 10.00)

INSERT
INTO MenuItemDetails
VALUES ('BigWay Hot Pot', 'Chinese', 0, 25.00)

INSERT
INTO MenuItemDetails
VALUES ('Roast Turkey', 'American', 0, 45.00)

INSERT
INTO MenuItemDetails
VALUES ('Beef Wellington', 'American', 0, 155.67)

INSERT
INTO MenuItemDetails
VALUES ('Mushroom Pizza', 'Italian', 0, 20.00)

INSERT
INTO MenuItemDetails
VALUES ('Beer', NULL, 0, 6.50)

INSERT
INTO MenuItemDetails
VALUES ('Milk Tea', 'Taiwanese', 0, 7.00)
```

INSERT
INTO MenuItemDetails
VALUES (‘Candied Bacon Bourbon’, NULL, 0, 18.00)

INSERT
INTO MenuItemDetails
VALUES (‘Lemonade’, NULL, 0, 3.25)

INSERT
INTO MenuItemDetails
VALUES (‘Root Beer, NULL, 0, 1.50)

Food

INSERT
INTO Food
VALUES (‘Garlic Beef’, ‘Entree’)

INSERT
INTO Food
VALUES (‘BigWay Hot Pot’, ‘Entree’)

INSERT
INTO Food
VALUES (‘Roast Turkey’, ‘Entree’)

INSERT
INTO Food
VALUES (‘Beef Wellington’ ‘Entree’)

INSERT
INTO Food
VALUES (‘Mushroom Pizza’, ‘’Entree’)

Beverages

INSERT
INTO Beverages

VALUES (‘Beer’, 1)

INSERT
INTO Beverages
VALUES (‘Milk Tea’, 0)

INSERT
INTO Beverages
VALUES (‘Candied Bacon Bourbon’, 1)

INSERT
INTO Beverages
VALUES (‘Lemonade’, 0)

INSERT
INTO Beverages
VALUES (‘Root beer’, 0)

AddOnPrice

INSERT
INTO AddOnPrice
VALUES (‘Fries’, 1.00)

INSERT
INTO AddOnPrice
VALUES (‘Mozzarella Sticks’, 1.50)

INSERT
INTO AddOnPrice
VALUES (‘Sakura Vanilla Ice Cream’, 0.00)

INSERT
INTO AddOnPrice
VALUES (‘Cranberry Sauce’, 1.50)

INSERT
INTO AddOnPrice
VALUES (‘Salad’, 2.00)

AddonCanHave

INSERT
INTO AddonCanHave
VALUES (‘Beef Wellington’, ‘Fries’)

INSERT
INTO AddonCanHave
VALUES (‘Mushroom Pizza’, ‘Mozzarella Sticks’)

INSERT
INTO AddonCanHave
VALUES (‘BigWay Hot Pot’, ‘Sakura Vanilla Ice Cream’)

INSERT
INTO AddonCanHave
VALUES (‘Roast Turkey’, ‘Cranberry Sauce’)

INSERT
INTO AddonCanHave
VALUES (‘Beef Wellington’, ‘Salad’)

Combo

INSERT
INTO Combo
VALUES (‘Beef Wellington and Candied Bacon Bourbon’, 195.58)

INSERT
INTO Combo
VALUES (‘BigWay Hot Pot and Milk Tea, 33.23)

INSERT
INTO Combo
VALUES (‘Mushroom Pizza and Lemonade’, 24.91)

INSERT
INTO Combo

VALUES (‘Roast Turkey and Beer’, 38.21)

INSERT
INTO Combo
VALUES (‘Mushroom Pizza and Root Beer’, 22.63)

PartOf

INSERT
INTO PartOf
VALUES (‘Beef Wellington and Candied Bacon Bourbon’, ‘Beef Wellington’)

INSERT
INTO PartOf
VALUES (‘Beef Wellington and Candied Bacon Bourbon’, ‘Candied Bacon Bourbon’’)

INSERT
INTO PartOf
VALUES (‘BigWay Hot Pot and Milk Tea’, ‘Milk Tea’’)

INSERT
INTO PartOf
VALUES (‘BigWay Hot Pot and Milk Tea’, ‘BigWay Hot Pot’)

INSERT
INTO PartOf
VALUES (‘Mushroom Pizza and Lemonade’, ‘Mushroom Pizza’)

INSERT
INTO PartOf
VALUES (‘Mushroom Pizza and Lemonade’, ‘Lemonade’)

INSERT
INTO PartOf
VALUES (‘Roast Turkey and Beer’, ‘Roast Turkey’)

INSERT
INTO PartOf

VALUES (‘Roast Turkey and Beer’, ‘Beer’)

INSERT
INTO PartOf
VALUES (‘Mushroom Pizza and Root Beer’, ‘Root Beer’)

INSERT
INTO PartOf
VALUES (‘Mushroom Pizza and Root Beer’, ‘Mushroom Pizza’)